

Scheduling Rigid Demands on Continuous-Time Linear Shift-Invariant Systems

Farhad Farokhi, Michael Cantoni, and Iman Shames

Abstract—We consider load scheduling on constrained continuous-time linear dynamical systems, such as automated irrigation and other distribution networks. The requested loads are rigid, i.e., the shapes cannot be changed. Hence, it is only possible to shift the order back-and-forth in time to arrive at a feasible schedule. We present a numerical algorithm based on using log-barrier functions to include the state constraints in the social cost function (i.e., an appropriate function of the scheduling delays). This algorithm requires a feasible initialization. Further, in another algorithm, we treat the state constraints as soft constraints and heavily penalize the constraint violations. This algorithm can even be initialized at an infeasible point. The applicability of both these numerical algorithms is demonstrated on an automated irrigation network with two pools and six farms.

I. INTRODUCTION

Scheduling problems arise in a variety of contexts. A peculiar scheduling problem is studied in this paper. It involves a constrained dynamical system and the processing of request to apply load, with a fixed but shiftable profile, on this system across time. The goal is to optimize a social measure of sensitivity to scheduling delay while satisfying hard constraints. This problem is motivated by an aspect of demand management in automated irrigation networks [1], [2], [3], and may arise in other areas. The main challenge associated with this problem relates to the rigidity of the load request, whereby the construction of a feasible schedule can only involve shifting requests back-and-forth in time. Relaxation of the rigidity requirement can lead to a formulation as a (large) linear program [1].

The formulation of the rigid load scheduling problem here distinguishes itself in the following ways. By contrast with [2], [3], a dynamics relationship between the load and the constrained system states are modelled. In [2], [3], only static capacity constraints are considered. The load scheduling problem considered in [1] does include dynamics, however this is modelled in discrete time. By contrast a continuous-time setting is employed in this paper. The discrete time formulation in [1] leads to a mixed-integer program, which is difficult to solve [4]. The continuous-time formulation here, on the other hand, gives rise to two gradient based numerical algorithms. The first involve log-barrier functions and thus a feasible initial point. The other uses a soft encoding of the state constraints, with heavy penalty on constraint violation, which does not require a feasible initial point. Both algorithms lead to only locally

optimal solutions due to the non-convexity of the scheduling problem.

The rest of the paper is organized as follows. We first formulate the problem in Section II. The numerical algorithms are presented in Sections V and Section IV. In Section V, the applicability of the developed algorithms is numerically studied on an automated irrigation network with two pools and six farms. Finally, Section VI concludes the paper.

II. PROBLEM FORMULATION

Consider the continuous-time linear time-invariant dynamical system

$$\dot{x}(t) = Ax(t) + Bu(t) + \sum_{i=1}^m E_i w_i(t), \quad x(0) = x_0, \quad (1)$$

where $x(t) \in \mathbb{R}^{n_x}$ is the state of the system, $u(t) \in \mathbb{R}^{n_u}$ is the control input (e.g., the water-level references in automated irrigation networks), and $w_i(t) \in \mathbb{R}^{n_{w,i}}$, $1 \leq i \leq m$, is a profile-constrained (e.g. on-off) input signal representing the scheduled load on the system corresponding to the supply of resources to customer i (e.g., the flow of the supplied water to each farmer in an irrigation network). Throughout this paper, we assume that the control signal $u(t)$ over the planning horizon $[0, T]$ with $T \in \mathbb{R}_{>0}$ is a discrete-time signal passed through a zero-order hold, that is, $u_i(t) = \alpha_{i,k}$ for all $1 \leq i \leq n_u$ and all $k\Delta \leq t < (k+1)\Delta$ with a given sampling time $0 < \Delta \leq T$. Although slightly conservative, this assumption allows us to work with finite-dimensional optimization problems instead of more complicated optimal control problems. For all integers $0 \leq k \leq K := \lceil T/\Delta \rceil - 1$ and $1 \leq i \leq n_u$, we define $\xi_{i,k}(t) = e_i[\text{step}(t - k\Delta) - \text{step}(t - (k+1)\Delta)]$, $\forall t \in [0, T]$, where the mapping $\text{step} : \mathbb{R} \rightarrow \{0, 1\}$ denotes the Heaviside step function, i.e., $\text{step}(t) = 1$ if $t \geq 0$ and $\text{step}(t) = 0$ otherwise. Moreover, $e_i \in \mathbb{R}^{n_u}$ is the column-vector with all entries equal to zero except the i -th entry which is equal to one. Therefore, we get $u(t) = u_0 + \sum_{k=1}^K \sum_{i=1}^{n_u} \alpha_{i,k} \xi_{i,k}(t)$, $\forall t \in [0, T]$, where $u_0 \in \mathbb{R}^{n_u}$ is the steady-state control input.

The customers submit demands $(v_i(t))_{t \in \mathbb{R}}$, $1 \leq i \leq m$. These demands are rigid (i.e., their shape cannot be changed). Hence, our decision variables are the delays that correspond to shifting the requested demand across the planning horizon, i.e., we select $\tau_i > 0$ so that $w_i(t) = v_i(t - \tau_i)$ for each $1 \leq i \leq m$. In doing so, the goal is to ensure that the state of the network $x(t)$ stays inside the feasible set $\mathcal{X} = \{x \in \mathbb{R}^{n_x} \mid Cx \leq d\}$. We can write this scheduling

This work was supported by the Australian Research Council (LP130100605), Rubicon Water Pty Ltd, and a McKenzie Fellowship.

The authors are with the Department of Electrical and Electronic Engineering, The University of Melbourne, Parkville, Victoria 3010, Australia. Emails: {ffarokhi, cantoni, ishames}@unimelb.edu.au

problem as

$$\min_{(\tau_i)_{i=1}^m, ((\alpha_i)_{i=1}^{n_u})_{k=1}^K} \sum_{i=1}^m h_i(\tau_i), \quad (2a)$$

$$\text{s.t.} \quad \underline{\tau}_i \leq \tau_i \leq \bar{\tau}_i, \forall i \in \{1, \dots, m\} \quad (2b)$$

$$\dot{x}(t) = Ax(t) + Bu(t) + \sum_{i=1}^m E_i v_i(t - \tau_i), x(0) = x_0, \quad (2c)$$

$$Cx(t) \leq d, \forall t \in [0, T], \quad (2d)$$

$$u(t) = \sum_{k=1}^K \sum_{i=1}^{n_u} \alpha_{i,k} \xi_{i,k}(t), \forall t \in [0, T], \quad (2e)$$

$$u_0 + \underline{u} \leq u(t) \leq u_0 + \bar{u}, \forall t \in [0, T], \quad (2f)$$

where $\underline{\tau}_i$ and $\bar{\tau}_i$ are the bounds on the scheduling delay for demand i , \underline{u} and \bar{u} are the bounds on the control signal deviations $u(t) - u_0$, and the continuously differentiable mapping $h_i : \mathbb{R} \rightarrow \mathbb{R}$ captures the sensitivity of customer i to the delay for scheduling its demand. Throughout the next section, we implicitly assume that T is long enough so that the optimization problem in (2) becomes feasible with a constant nominal control input (i.e., if the demands are separated from each other “to some degree”, the state of the system stays feasible without any effort). This assumption is made to make sure that we can always find a feasible initial condition for the numerical algorithm, proposed in the next section, by simply separating the demands from each other. Towards the end of this paper, we present another approach for solving our scheduling problem that avoids requiring a feasible initial condition by treating the constraints on the state as soft constraints.

III. NUMERICAL ALGORITHM

In this section, we present a numerical algorithm for solving (2) by adding the state constraints in (2d) to the cost function using log-barrier functions. Let us define

$$\bar{x}_0(t) = \exp(At)x_0 + \int_0^t \exp(A(t-\beta))Bu_0 d\beta,$$

$$\bar{x}_{i,k}^u(t) = \int_0^t \exp(A(t-\beta))B\xi_{i,k}(\beta) d\beta, \forall i \in \{1, \dots, n_u\}, \quad \forall k \in \{1, \dots, K\},$$

$$\bar{x}_i^v(t) = \int_0^t \exp(A(t-\beta))E_i v_i(\beta) d\beta, \forall i \in \{1, \dots, m\}.$$

Since the underlying system in (1) is linear and time invariant, the solution of the ordinary differential equation (1) can be written explicitly as $x(t) = \bar{x}_0(t) + \sum_{k=1}^K \sum_{i=1}^{n_u} \alpha_{i,k} \bar{x}_{i,k}^u(t) + \sum_{i=1}^m \bar{x}_i^v(t - \tau_i)$. Now, we can rewrite the optimization problem in (2) as

$$\min_{(\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K} \sum_{i=1}^m h_i(\tau_i), \quad (3a)$$

$$\text{s.t.} \quad x(t) = \bar{x}_0(t) + \sum_{k=1}^K \sum_{i=1}^{n_u} \alpha_{i,k} \bar{x}_{i,k}^u(t) + \sum_{i=1}^m \bar{x}_i^v(t - \tau_i), \quad (3b)$$

$$Cx(t) \leq d, \forall t \in [0, T], \quad (3c)$$

$$\underline{\tau}_i \leq \tau_i \leq \bar{\tau}_i, \forall i \in \{1, \dots, m\}, \quad (3d)$$

$$\underline{u}_i \leq \alpha_{i,k} \leq \bar{u}_i, \forall i \in \{1, \dots, n_u\}, \quad \forall k \in \{1, \dots, K\}. \quad (3e)$$

This optimization problem is still difficult to solve as we have to check infinitely many constraints; see (3c). Let us use the notation C_j , $1 \leq j \leq p$, to denote the rows of the matrix $C \in \mathbb{R}^{p \times n_x}$. We add the state constraints in (3c) to the cost function using log-barrier functions. This transforms the optimization problem in (3) to

$$\min_{(\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K} J((\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) \quad (4a)$$

$$\text{s.t.} \quad \underline{\tau}_i \leq \tau_i \leq \bar{\tau}_i, \forall i \in \{1, \dots, m\}, \quad (4b)$$

$$\underline{u}_i \leq \alpha_{i,k} \leq \bar{u}_i, \forall i \in \{1, \dots, n_u\}, \quad \forall k \in \{1, \dots, K\}, \quad (4c)$$

where

$$\begin{aligned} J((\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) &= \sum_{i=1}^m h_i(\tau_i) - \sum_{z=1}^p \int_0^T \epsilon \log \left(-C_z \left[\bar{x}_0(t) \right. \right. \\ &\quad \left. \left. + \sum_{k=1}^K \sum_{i=1}^{n_u} \alpha_{i,k} \bar{x}_{i,k}^u(t) + \sum_{i=1}^m \bar{x}_i^v(t - \tau_i) \right] + d_z \right) dt \end{aligned}$$

in which $\epsilon \in \mathbb{R}_{>0}$ is an appropriately selected parameter.

Remark 1: With increasing ϵ , the optimal solution is pushed further from the boundary of the feasible set. Therefore, to recover the optimal scheduling, we need to sequentially reduce ϵ and employ the solution of each step as the initialization of the next step. This would result in a more numerically stable algorithm; see the log barrier methods in [5].

Lemma 1: $J((\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K)$ is a continuously differentiable function. Moreover,

$$\begin{aligned} \frac{\partial}{\partial \tau_\ell} J((\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) &= \frac{d}{d\tau_\ell} h_\ell(\tau_\ell) \\ &\quad + \sum_{z=1}^p \int_0^T \epsilon \frac{-C_z(A\bar{x}_\ell^v(t - \tau_\ell) + E_\ell v_\ell(t - \tau_\ell))}{-C_z x(t; (\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) + d_z} dt \\ \frac{\partial}{\partial \alpha_{j,\ell}} J((\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) &= \sum_{z=1}^p \int_0^T \epsilon \frac{C_z \bar{x}_{j,\ell}^u(t)}{-C_z x(t; (\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) + d_z} dt \end{aligned}$$

where

$$\begin{aligned} x(t; (\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) &= \bar{x}_0(t) + \sum_{k=1}^K \sum_{i=1}^{n_u} \alpha_{i,k} \bar{x}_{i,k}^u(t) + \sum_{i=1}^m \bar{x}_i^v(t - \tau_i). \end{aligned}$$

Proof: First note that

$$\begin{aligned} \frac{\partial}{\partial \tau_\ell} \int_0^T \epsilon \log(-C_z x(t; (\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) + d_z) dt &= \int_0^T \epsilon \frac{-C_z \partial x(t; (\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) / \partial \tau_\ell}{-C_z x(t; (\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) + d_z} dt \end{aligned}$$

where

$$\begin{aligned} \frac{\partial}{\partial \tau_\ell} x(t; (\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) \\ = \frac{\partial}{\partial \tau_\ell} \bar{x}_\ell^v(t - \tau_\ell) \\ = -\dot{\bar{x}}_\ell^v(t - \tau_\ell) \\ = -(A\bar{x}_\ell^v(t - \tau_\ell) + E_\ell v_\ell(t - \tau_\ell)). \end{aligned}$$

Similarly, we have

$$\begin{aligned} \frac{\partial}{\partial \alpha_{j,\ell}} \int_0^T \epsilon \log(-C_z x(t; (\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) + d_z) dt \\ = \int_0^T \epsilon \frac{-C_z \partial x(t; (\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) / \partial \alpha_{j,\ell}}{-C_z x(t; (\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) + d_z} dt \end{aligned}$$

where

$$\frac{\partial}{\partial \alpha_{j,\ell}} x(t; (\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) = \bar{x}_{j,\ell}^u(t).$$

The rest of the proof follows from simple algebraic manipulations. ■

Now, we can use Algorithm 1 (overleaf) to recover a local solution of (4). We can select the step sizes $\mu_l^{\tau_i}$ and $\mu_l^{\alpha_{j,\ell}}$ using backtracking line search algorithm [5, p.464] and terminate the algorithm whenever the improvements in the cost function becomes negligible. Unfortunately, this algorithm requires a feasible starting point (because the argument of the logarithmic functions cannot become negative). We remove this assumption in the next section by proposing a numerical procedure that treats the state constraints as soft constraints.

IV. SOFT CONSTRAINTS ON STATES

In the previous section, we were required to find a feasible initialization to be able to run Algorithm 1. Here, we take a different approach by solving the optimization problem

$$\min_{(\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K} \sum_{i=1}^m h_i(\tau_i) + \sum_{z=1}^p \int_0^T e^{\vartheta(C_z x(t) - d_z)} dt, \quad (5a)$$

$$\text{s.t.} \quad \underline{\tau}_i \leq \tau_i \leq \bar{\tau}_i, \forall i \in \{1, \dots, m\} \quad (5b)$$

$$\dot{x}(t) = Ax(t) + Bu(t) + \sum_{i=1}^m E_i v_i(t - \tau_i), \quad (5c)$$

$$x(0) = x_0, \quad (5d)$$

$$u(t) = \sum_{k=1}^K \sum_{i=1}^{n_u} \alpha_{i,k} \xi_{i,k}(t), \forall t \in [0, T], \quad (5e)$$

$$\underline{u} \leq u(t) \leq \bar{u}, \forall t \in [0, T], \quad (5f)$$

where $\vartheta \in \mathbb{R}_{>0}$ is an appropriately selected constant. In this problem, we may violate the constraints $Cx(t) - d \leq 0$, however, the term $\sum_{z=1}^p \int_0^T e^{\vartheta(C_z x(t) - d_z)} dt$ heavily penalizes such violations. For small values of ϑ , this term also penalizes the states being close to the boundary (of the feasible set), however, as we increase ϑ , this term approaches zero inside the feasible set and infinity outside of the feasible set.

Note that similar to the previous section, we can transform (5) into

$$\min_{(\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K} \sum_{i=1}^m h_i(\tau_i) + \sum_{z=1}^p \int_0^T e^{\vartheta(C_z x(t) - d_z)} dt, \quad (6a)$$

$$\text{s.t.} \quad x(t) = \bar{x}_0(t) + \sum_{k=1}^K \sum_{i=1}^{n_u} \alpha_{i,k} \bar{x}_{i,k}^u(t) + \sum_{i=1}^m \bar{x}_i^v(t - \tau_i), \quad (6b)$$

$$\underline{u}_i \leq \alpha_{i,k} \leq \bar{u}_i, \forall i \in \{1, \dots, n_u\}, \quad \forall k \in \{1, \dots, K\}. \quad (6c)$$

Let us define

$$\begin{aligned} J'((\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) \\ = \sum_{i=1}^m h_i(\tau_i) + \sum_{z=1}^p \int_0^T \exp\left(\vartheta\left(C_z \left[\bar{x}_0(t) + \sum_{k=1}^K \sum_{i=1}^{n_u} \alpha_{i,k} \bar{x}_{i,k}^u(t) + \sum_{i=1}^m \bar{x}_i^v(t - \tau_i)\right] - d_z\right)\right) dt. \end{aligned}$$

Hence, we may rewrite (6) as

$$\min_{(\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K} J'((\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K), \quad (7a)$$

$$\text{s.t.} \quad \underline{\tau}_i \leq \tau_i \leq \bar{\tau}_i, \forall i \in \{1, \dots, m\}, \quad (7b)$$

$$\underline{u}_i \leq \alpha_{i,k} \leq \bar{u}_i, \forall i \in \{1, \dots, n_u\}, \quad \forall k \in \{1, \dots, K\}. \quad (7c)$$

Similarly, we can prove the following result regarding the augmented cost function.

Lemma 2: $J'((\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K)$ is a continuously differentiable function. Moreover,

$$\begin{aligned} \frac{\partial}{\partial \tau_\ell} J'((\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) &= \frac{d}{d\tau_\ell} h_\ell(\tau_\ell) \\ &- \sum_{z=1}^p \int_0^T \vartheta \exp(\vartheta(C_z x(t; (\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) - d_z)) \\ &\quad \times C_z (A\bar{x}_\ell^v(t - \tau_\ell) + E_\ell v_\ell(t - \tau_\ell)) dt, \\ \frac{\partial}{\partial \alpha_{j,\ell}} J'((\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) \\ &= \sum_{z=1}^p \int_0^T \vartheta \exp(\vartheta(C_z x(t; (\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) - d_z)) \\ &\quad \times C_z \bar{x}_{j,\ell}^u(t) dt, \end{aligned}$$

where $x(t; (\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K)$ is defined as in Lemma 1.

Proof: First, note that

$$\begin{aligned} \frac{\partial}{\partial \tau_\ell} \int_0^T \exp(\vartheta(C_z x(t; (\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) - d_z)) dt \\ = \int_0^T \vartheta \exp(\vartheta(C_z x(t; (\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) - d_z)) \\ \times C_z \left[\frac{\partial}{\partial \tau_\ell} x(t; (\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) \right] dt. \end{aligned}$$

Algorithm 1 Projected gradient algorithm for scheduling rigid demands.

Require: Feasible initialization $(\tau_i[0])_{i=1}^m$ and $((\alpha_{i,k}[0])_{i=1}^{n_u})_{k=1}^K$

1: **for** $l = 1, 2, \dots$ **do**
2: Update

$$\tau_\ell[l] = P_{\underline{\tau}_\ell}^{\bar{\tau}_\ell} \left[\tau_\ell[l-1] - \mu_l^{\tau_i} \frac{\partial J((\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K)}{\partial \tau_i} \right]_{((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K = ((\alpha_{i,k}[l-1])_{i=1}^{n_u})_{k=1}^K}, \forall \ell \in \{1, \dots, m\},$$

and

$$\alpha_{j,\ell}[l] = P_{\underline{\alpha}_{j,\ell}}^{\bar{\alpha}_{j,\ell}} \left[\alpha_{j,\ell}[l-1] - \mu_l^{\alpha_{j,\ell}} \frac{\partial J((\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K)}{\partial \alpha_{j,\ell}} \right]_{((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K = ((\alpha_{i,k}[l-1])_{i=1}^{n_u})_{k=1}^K}, \forall j \in \{1, \dots, n_u\},$$

$$\forall \ell \in \{1, \dots, K\},$$

where, for constants $\beta < \gamma$, $P_\beta^\gamma[x] = \beta$ if $x < \beta$, $P_\beta^\gamma[x] = x$ if $\beta \leq x \leq \gamma$, and $P_\beta^\gamma[x] = \gamma$ if $x > \gamma$.

3: **end for**

TABLE I
NUMERICAL PARAMETERS USED IN THE SIMULATION.

	$c_{in,i}$	$c_{out,i}$	$t_{d,i}$	κ_i	ϕ_i	ρ_i
$i = 1$	0.0546	0.0363	5	0.0103	71.820	8.510
$i = 2$	0.0173	0.0258	6	0.0084	141.27	16.74

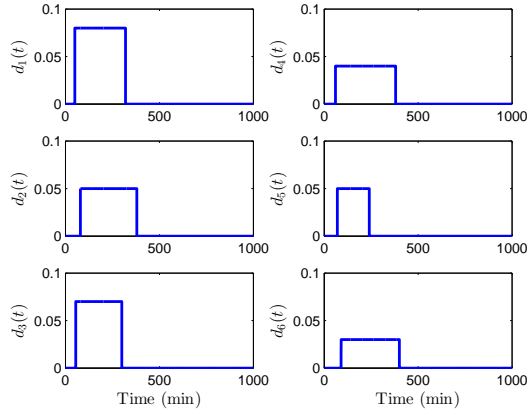


Fig. 1. The demand by the farms as requested (without the scheduling delays).

Similarly, we have

$$\begin{aligned} & \frac{\partial}{\partial \alpha_{j,\ell}} \int_0^T \exp(\vartheta(C_z x(t; (\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) - d_z)) dt \\ &= \int_0^T \vartheta \exp(\vartheta(C_z x(t; (\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) - d_z)) \\ & \quad \times C_z \left[\frac{\partial}{\partial \alpha_{j,\ell}} x(t; (\tau_i)_{i=1}^m, ((\alpha_{i,k})_{i=1}^{n_u})_{k=1}^K) \right] dt. \end{aligned}$$

The rest of the proof follows from simple algebraic manipulations. ■

Algorithm 1 may be used with the gradients in Lemma 2 to find a local solution of the optimization problem in (7). Moreover if, after finding the optimal solution for a given ϑ , the state constraints were violated at an intolerable level, we may sequentially increase ϑ and solve the problem until we get acceptable performance.

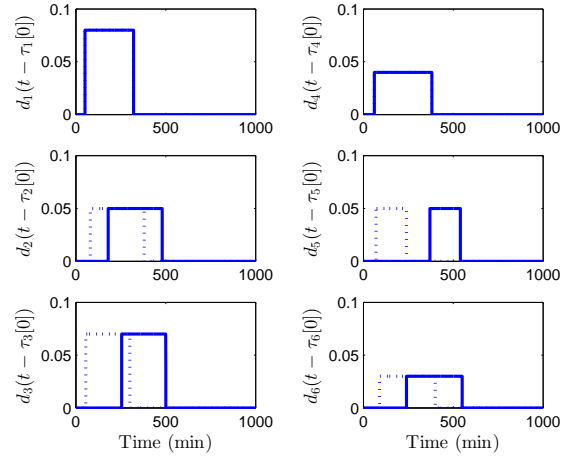


Fig. 2. The shifted demands at the initialization of the algorithm. The dotted curve demonstrates the requests and the solid curve demonstrates their shifted counterpart.

V. NUMERICAL EXAMPLE

In this section, we illustrate the applicability of the algorithms on a water channel with two pools. The numerical example is borrowed from [1]. Each pool is modelled as

$$y_i(s) = \frac{c_{in,i}}{s} e^{-t_{d,i}s} q_i(s) - \frac{c_{out,i}}{s} q_{i+1}(s) - \frac{c_{out,i}}{s} \zeta_i(s),$$

where $c_{in,i}$ and $c_{out,i}$ are discharge rates determined by the physical characteristics of the gates used to set the flow between neighbouring pools, and $t_{d,i}$ is the delay associated with the transport of water along the pool. Here, $\zeta_i(s)$ denotes the overall off-take flow load on pool i , that is, all the water supplied to the farms connected to this pool. Moreover, $q_i(s)$ is the flow of water from pool $i-1$ to pool i and $y_i(s)$ denotes the water level in pool i . For the purpose of this example, we replace the delays with their first-order Padé approximation¹. Each pool is controlled, locally, by

$$q_i(s) = \frac{\kappa_i(\phi_i s + 1)}{s(\rho_i s + 1)} (u_i(s) - y_i(s)),$$

¹Note that the choice of a first-order Padé approximation is justifiable as the pool delays are all parts of closed-loops (with local controllers), with loop-gain cross-overs that are sufficiently small to make the overall closed-loop behaviour insensitive to the approximation error [6].

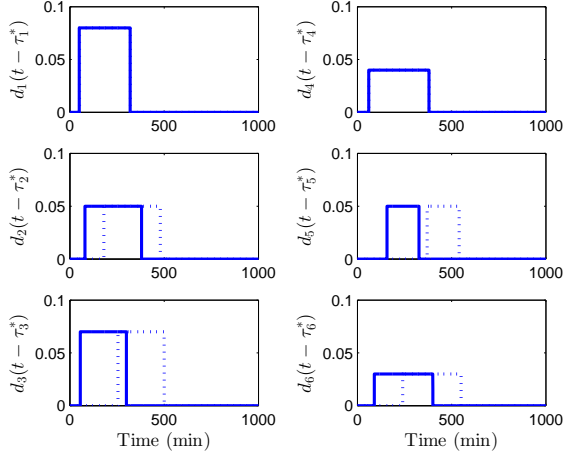


Fig. 3. The shifted demands for the local solution recovered by Algorithm 1 with $\epsilon = 0.1$. The dotted curve demonstrates the shifted demands at the initialization and the solid curve demonstrates the shifted demands at the locally optimal solution.

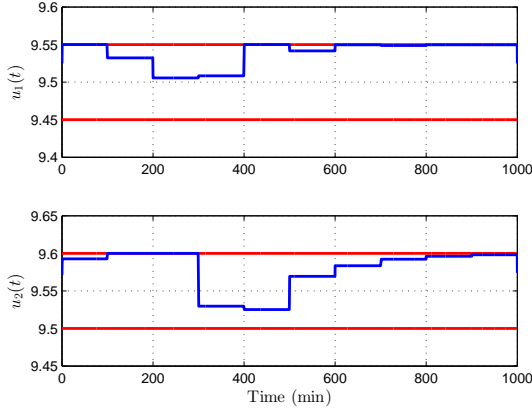


Fig. 4. The reference signal for the local solution recovered by Algorithm 1 with $\epsilon = 0.1$. The red lines show the boundary of the feasible region.

where κ_i , ϕ_i , and ρ_i are appropriately selected control parameters. Furthermore, $u_i(s)$ denotes the water-level reference signal of pool i . Table I shows the parameters used in this example. The state constraints are as follows $9.4 \leq y_1(t) \leq 9.7$ and $9.5 \leq y_2(t) \leq 9.7$. Finally, throughout this example, we fix $u_0 = [9.50, 9.55]^\top$.

Figure 1 illustrates the requested demands of the farms. Here, $(v_i(t))_{i=1}^3$ and $(v_i(t))_{i=4}^6$, respectively, denote demands for pool 1 and 2. Let us select linear penalty functions $h_i(\tau_i) = \tau_i$ for all i . Moreover, assume that the reference signal should belong to a bounded region captured by

$$u_0 - \begin{bmatrix} 0.05 \\ 0.05 \end{bmatrix} \leq u(t) \leq u_0 + \begin{bmatrix} 0.05 \\ 0.05 \end{bmatrix}.$$

Note that without these control input constraints, one can schedule all the loads without any delay but with large control input deviations. In this example, we select $\tau_i = 0$, $\forall i$, which means that we can only shift demands forward. Let us also select $\bar{\tau}_i = 300$ min for all i .

First, we use Algorithm 1 to extract a reasonable schedule by shifting these demands. This algorithm requires a feasible starting point, which can be constructed by shifting the

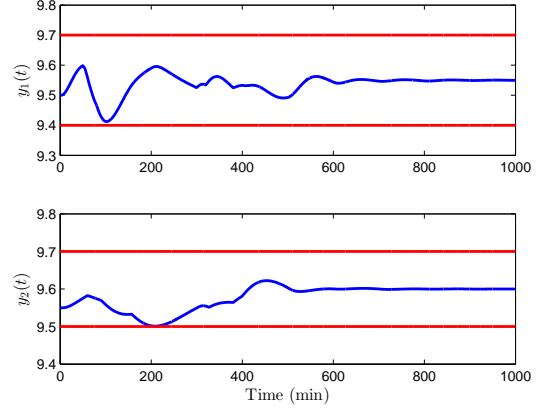


Fig. 5. The outputs for the local solution recovered by Algorithm 1 with $\epsilon = 0.1$. The red lines show the boundary of the feasible region.

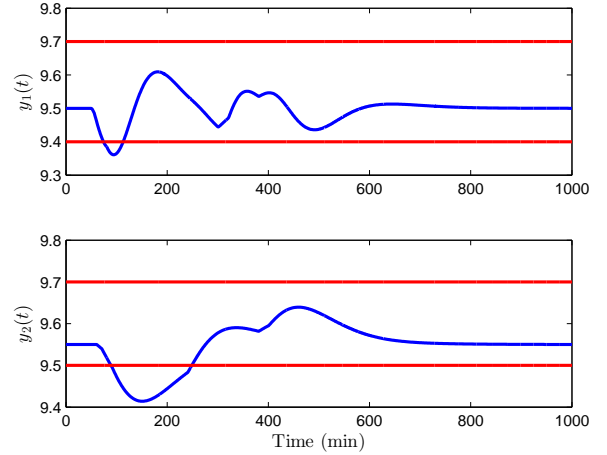


Fig. 6. The output of the system when all the decision variables (the control inputs and scheduling delays) are set equal to zero.

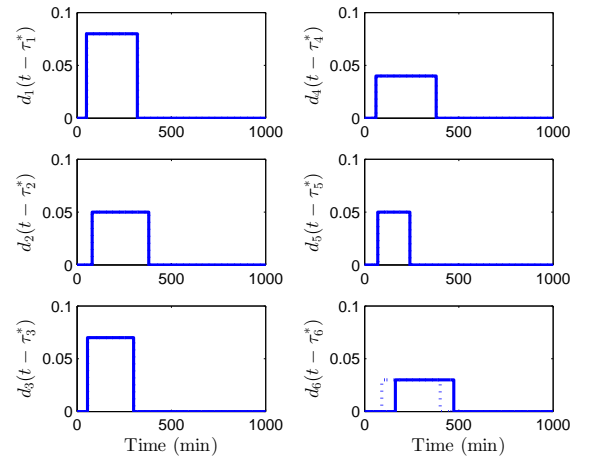


Fig. 7. The shifted demands for the local solution recovered by the proposed algorithm in Section IV with $\vartheta = 100$. The dotted curve demonstrates the shifted demands at the initialization and the solid curve demonstrates the shifted demands at the suboptimal solution.

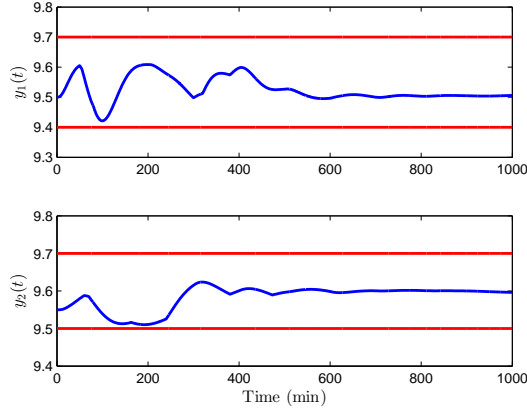


Fig. 8. The outputs for the locally optimal solution recovered by the proposed algorithm in Section IV with $\vartheta = 100$. The red lines show the boundary of the feasible region.

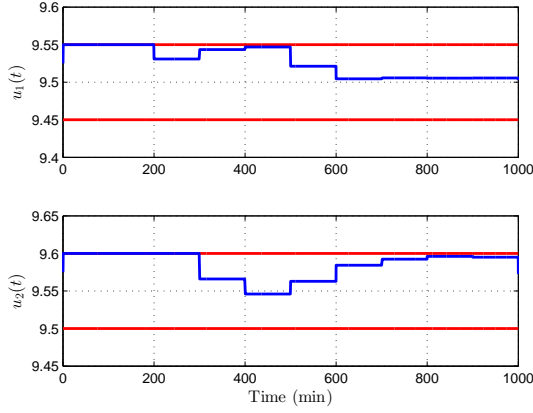


Fig. 9. The reference signal for the locally optimal solution recovered by the proposed algorithm in Section IV with $\vartheta = 100$. The red lines show the boundary of the feasible region.

demands (to be somewhat distant from each other). Figure 2 illustrates the shifted demands at this initialization (solid curve) as well as the original requests (dotted curves) for comparison. Let us fix $\epsilon = 0.1$. Figure 3 shows the shifted demands for the local solution recovered by the proposed algorithm. As we expect, the delays are significantly smaller in comparison to the initialization. Figure 4 portrays the reference signals for the solution and Figure 5 illustrates the outputs. Evidently, the output stays in the desired region. Although powerful, Algorithm 1 requires a feasible initial condition that may not be easy to find. Therefore, in the rest of this section, we study the method presented in Section IV.

Let us select $\vartheta = 100$. Figure 7 illustrates the shifted demands for the locally optimal solution recovered by the proposed algorithm. At the starting point (fed to the algorithm), all the decision variables are selected to be equal to zero for which the state of the system does not stay in the feasible region; see Figure 6. Figures 8 and 9 show the output and the control for the local solution recovered by the proposed algorithm in Section IV. Interestingly, all the constraints are satisfied, which is because of the large value of ϑ . If we reduce ϑ to be equal to 10, the outputs violate the constraints on the state as shown in Figure 10. Evidently, by

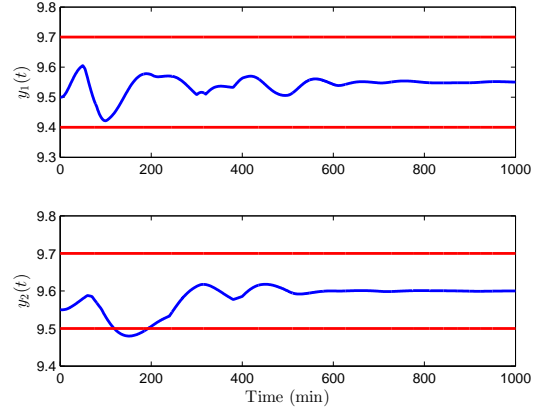


Fig. 10. The outputs for the locally optimal solution recovered by the proposed algorithm in Section IV with $\vartheta = 10$. The red lines show the boundary of the feasible region.

comparing Figures 6, 8, and 10, we can see that by increasing ϑ , the constraint violations are becoming more infrequent (until they do not occur at all).

VI. CONCLUSIONS

In this paper, we presented numerical algorithms for scheduling demands on continuous-time linear time-invariant systems. The rigidity of the demands dictated that we can only shift them back-and-forth in time (and cannot change their shapes). The first algorithm used log-barrier functions to include the state constraints in the cost function. The second algorithm considered the state constraints as soft constraints and added a penalty function for the constraint violations to the cost function. Future research can focus on constructing a market mechanism for achieving the optimal schedule based on the customers preferences. We can also compute an optimality gap via finding a lower-bound for the solution of the dual of the problem corresponding to (2).

REFERENCES

- [1] J. Alende, Y. Li, and M. Cantoni, "A $\{0, 1\}$ linear program for fixed-profile load scheduling and demand management in automated irrigation channels," in *Proceedings of the 48th IEEE Conference on Decision and Control held jointly with the 28th Chinese Control Conference*, 2009, pp. 597–602.
- [2] S. Hong, P.-O. Malaterre, G. Belaud, and C. Dejean, "Optimization of irrigation scheduling for complex water distribution using mixed integer quadratic programming (MIQP)," in *Proceedings of the 10th International Conference on Hydroinformatics (HIC 2012)*, 2012.
- [3] J. M. Reddy, B. Wilamowski, and F. Cassel-Sharmasarkar, "Optimal scheduling of irrigation for lateral canals," *ICID Journal on Irrigation and Drainage*, vol. 48, no. 3, pp. 1–12, 1999.
- [4] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, ser. Series of Books in the Mathematical Sciences. W. H. Freeman, 1979.
- [5] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
- [6] M. Cantoni, E. Weyer, Y. Li, S. K. Ooi, I. Mareels, and M. Ryan, "Control of large-scale irrigation networks," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 75–91, 2007.